---------------------------- CEN 4072/6070 Software Testing & Verification --------------------------

Exam 1 – Spring 2017 – Solution Notes

1. **Validation testing** sometimes refers to testing undertaken to *demonstrate that a system performs correctly*. **Defect testing** sometimes refers to testing undertaken to *expose system defects.*

2. a

3. G, K, C, (E or F), A, D, B, I

4. a. v; b. vi; c. v; d. iv; e. i; f. v

5. E, A, K, B, A, H, G, C, K, B, F, E

6. An operational profile describes how a system is expected to be used in a particular end-user environment.

7. c

8. f

9. a. Minimum Viable Quality (MVQ). It is based (and builds) on the premise that some companies test their web-based software services too much before releasing them to production, since speed of release is the vital competitive advantage in the world of web based software services.

   b. i. Big Up-Front Testing. The expensive and time-consuming testing traditionally applied to software *prior to release* to reduce and remove defects and meet the end-customers' needs. The alternative to BUFT that Ken proposes is MVQ.

   ii. Last Known Good Configuration. If the quality of a new software component is found to be unacceptable, then a quick failback to the Last Known Good Configuration is effected in MVQ.

   iii. The lines added, modified, or deleted from a file from one version to another – it is a measure of the changes made to a software component over a period of time. Ken argued that the risks associated with code churn per release are reduced when code deployment is *continuous*.

   iv. *Feature flighting*, also known as feature toggling or feature switching, is a technique in which a feature or set of feature-related changes can be *turned on or off (remotely)* without dependency on other code. Flighting directly supports MVQ by allowing for both evaluating the quality of new features and experimentation

10. f

11. a. minimum # of cases needed for *strong equivalence class testing:* 16
    minimum # of cases needed for *weak equivalence class testing:* 4
    b. minimum # of cases needed to cover the partitions: 4

12. The problem is with measures such as "number of errors" or "number of bugs" being ill-defined and therefore difficult to interpret objectively. The program, which had no body, makes it clear that such measures can be difficult to interpret.

13. a. When encountering a 3rd-degree AND-node that must be False, the rule allows us to ignore (“cull”) all combinations of input values such that more than one input is False. Thus, we consider only those 3 combinations of input condition for which exactly one input value is False. This reduces the number of combinations to consider from 7 to 3.
    b. The rationale is that the most important cases to explore are probably those associated with the **minimally sufficient conditions** – i.e., exactly one incoming False edge for the AND node to evaluate to False.

14. 6
    16:4 = 40:X, so X=10. Since 4 of the 10 “real” bugs have already been discovered, there are about 6 bugs still to be found. Remaining non-seeded: X-x = x(N/n – 1) = 4(40/16 – 1) = 160/16 - 4 = 10-4 = 6

15. d

16. a. true; b. false; c. true; d. true; e. false

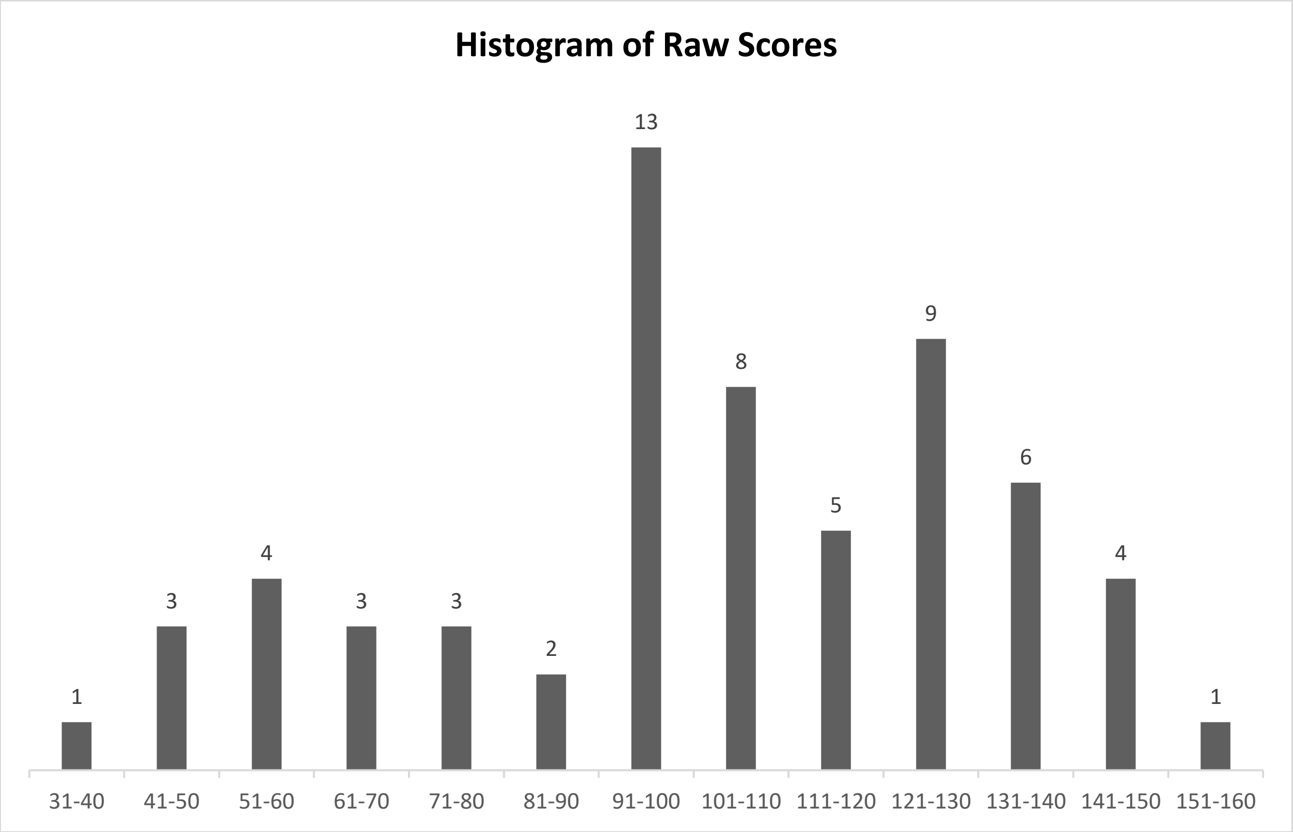
17. d

18. a. true; b. false; c. false; d. true; e. true

19. e

20. a. is not; b. is; c. is; d. is not; e. is not; f. is; g. is not; h. is; i. is; j. is not

21. F, C, K, N, P, B, R, H, I, J, G, L, X, H, T, W, M, O


Histogram of Raw Scores